Attorney Docket No.: 17887-005320US

PATENT APPLICATION

Mobile Software Morphing Agent

Inventors:

Greg I. Chiou, a citizen of the United States, residing at,
19388 Miller Ct.
Saratoga, California 95070

Lev Stesin, a citizen of the United States, residing at,
270 26th Avenue #10
San Francisco, California 94121

Arup Mukherjee, a citizen of Canada, residing at,
110 Harbor Seal Ct.
San Mateo, California 94404

09502773.082900

Assignee:

Yahoo! Inc.
3420 Central Expressway
Santa Clara, CA 95051

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 415-576-0200

Mobile Software Morphing Agent

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is related to U.S. Provisional Patent Application Serial No. 60/212,060 (Atty. Docket No. 017887-005300), filed June 16, 2000, entitled "Mobile Software Morphing Agent," the disclosure of which is hereby incorporated by reference in its entirety.

Ins A1

BACKGROUND OF THE INVENTION

The present invention relates generally to modifying and translating information received from a remote source, and more particularly to modifying and translating executable code and data received from a web site.

The World Wide Web (WWW), or "the Web", is now the premier outlet to publish information of all types and forms. Documents published on the web, commonly called Web pages, are typically published using a markup language such as HTML (or Hyper Text Markup Language), which sets standards for the formatting of documents. Additionally, These standards make it possible for people to read and understand documents no matter which program they use for that purpose. Often included in an HTML formatted web page are software code segments attached as part of the page. Examples of such software include JavaScript, Java and ActiveX commands. If a user's browser is enabled to process the software code, the code will typically be processed to provide additional windows, e.g., pop-up windows, forms and other content for presentation to the user.

Typically, a user accesses pages on the Web through a web portal. One common portal is Yahoo located at URL: http://www.yahoo.com/. When a user selects a reference such as a URL presented on a page provided by the portal, the users browser will access another page associated with the URL at a remote site. From then on, the user will be connected to the remote server unless the browser is instructed to return to the portal (e.g., via a "back" button or a "home" button displayed on the browser). In the commerce context, for example, a user may access a remote commerce site and conduct transactions, e.g., to purchase a product. In this case, the portal is completely unaware of any transactions or information exchange between the user and the remote site.

It is therefore desirable for a web portal to provide a page from a remote site, such as a remote commerce site, via a special proxy server to a user and to keep the user connected to the proxy so that information exchange between the user and remote server can be monitored by the proxy. However, it may be necessary to modify HTML formatting, HTML links and JavaScript code associated with a page provided by a remote site so that information exchange activity is directed to the proxy. For example, it is desirable to translate a link directed to a particular site into a link directed to the proxy so that the proxy handles access to the desired page from the particular site.

Accordingly, it is desirable to provide a configurable system to parse and translate downloadable software and/or content without additional development efforts from the original software and content provider. Additionally, it is desirable to provide an adaptive system to serve a corresponding software morphing agent to handle the original software and content.

SUMMARY OF THE INVENTION

The present invention provides systems and methods for extending or modifying the behavior of executable code and/or data that can be downloaded to a client device (e.g., a PC, laptop, PalmPilot, PDA or other hand-held communication device, etc.) implementing a browser program (e.g., Microsoft Internet Explorer, Netscape Navigator, a microbrowser such as a WAP enabled browser, etc.). The present invention is particularly useful for modifying web content, such as a web page received from a web site, including JavaScript code and/or HTML data.

According to the invention, one or more morphing agents are provided for translating and modifying code and data from a software source, such as a remote server. Each morphing agent translates and modifies a particular type of code. For example, one morphing agent may be provided for processing JavaScript code and another may be provided for processing HTML code and data. It will be appreciated that one morphing agent may be provided for processing multiple types of code, for example, one morphing agent for processing JavaScript and HTML code. Each morphing agent typically includes a tokenizer module, a parser module and a translation module, each of which implements specific rule sets. Original software content is first tokenized according to a set of tokenizer rules, and subsequently parsed according to a set of parser rules to determine relationships between the tokens. The parsed code is then translated according to the set of translator rules to produce the desired modified software content. An exception handler module is also provided for implementing exception rules when an exception occurs.

In operation, a user establishes a connection with a proxy server using the browser program on the client device, and the proxy server establishes a connection with the software source. The original software content is downloaded by the proxy server. All modules of a particular morphing agent can be located either on the client device or on the proxy server, or they may be spread between the client device and proxy server. Thus, if all modules reside on the proxy server, the morphing agent modifies the original software content and the modified content is downloaded to the client device. Similarly, if all modules reside on the client device, the original content is downloaded to the client device for processing by the morphing agent at the client device. If some of the modules reside on the proxy server, those module process the original content and the partially processed code is downloaded to the client device for processing by the remaining modules.

According to an aspect of the present invention, a computer implemented method is provided for modifying code to be compatible with a runtime library, wherein the code is received from a remote source. The method typically comprises the steps of receiving the code segment from the remote source, tokenizing the code segment into a plurality of tokens, and parsing the plurality of tokens so as to determine relationships between the plurality of tokens. The method also typically includes the step of translating the code segment into a modified code segment based on the determined relationships between the tokens such that the modified code segment is compatible with the runtime library.

According to another aspect of the present invention, a computer readable medium containing instructions for controlling a computer system to modify a code segment received from a remote source to be compatible with a runtime library is provided. The instructions typically include instructions to tokenize the code segment into a plurality of tokens, and parse the plurality of tokens so as to determine relationships between the plurality of tokens. The instructions also typically include instructions to translate the code segment into a modified code segment based on the determined relationships between the tokens such that the modified code segment is compatible with the runtime library.

Reference to the remaining portions of the specification, including the drawings and claims, will realize other features and advantages of the present invention. Further features and advantages of the present invention, as well as the structure and operation of various embodiments of the present invention, are described in detail below with respect to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a general overview of an information retrieval and communication network including a proxy server, client devices, and remote servers according to an embodiment of the present invention;

Figure 2 illustrates various configurations of a runtime environment of a morphing agent according to embodiments of the present invention; and

Figure 3 illustrates a general processing flow between modules of a morphing agent according to an embodiment of the present invention.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

Figure 1 illustrates a general overview of an information retrieval and communication network 10 including a proxy server 20, clients 30_1 to 30_N, and remote servers 50_1 to 50_N according to an embodiment of the present invention. In computer network 10, clients 30_1 to 30_N are coupled through the Internet 40, or other communication network, to proxy server 20 and servers 50_1 to 50_N. Only one proxy server 20 is shown, but it is understood that more than one proxy server can be used and that other servers providing additional functionality may also be interconnected to any component shown in network 10 either directly, over a LAN or a WAN, or over the Internet.

Several elements in the system shown in Figure 1 are conventional, well-known elements that need not be explained in detail here. For example, each client 30 could be a desktop personal computer, workstation, cellular telephone, personal digital assistant (PDA), laptop, or any other computing device capable of interfacing directly or indirectly to the Internet. Each client 30 typically runs a browsing program, such as Microsoft's Internet Explorer, Netscape Navigator, or a WAP enabled browser in the case of a cell phone, PDA or other wireless device, or the like, allowing a user of client 30 to browse pages available to it from proxy server 20, servers 50_1 to 50_N or other servers over Internet 40. Each client 30 also typically includes one or more user interface devices 32, such as a keyboard, a mouse, touchscreen, pen or the like, for interacting with a graphical user interface (GUI) provided by the browser on a monitor screen, LCD display, etc., in conjunction with pages and forms provided by proxy server 20, servers 50_1 to 50_N or other servers. The present invention is suitable for use with the Internet, which refers to a specific global Internetwork of networks. However, it should be understood that other networks can be used instead of the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

According to one embodiment as will be described in more detail below, proxy server 20 and/or clients 30, and all of their related components are operator configurable using an application including computer code run using a central processing unit such as an Intel Pentium processor or the like. Computer code for operating and configuring proxy server 20 and/or clients 30 as described herein is preferably stored on a hard disk, but the entire program code, or portions thereof, may also be stored in any other memory device such as a ROM or RAM, or provided on any media capable of storing program code, such as a compact disk medium, a DVD, a floppy disk, or the like. Additionally, the entire program code, or portions thereof may be downloaded to clients 30 or otherwise transmitted as is well known, e.g., from proxy server 20 over the Internet, or through any other conventional network connection as is well known, e.g., extranet, VPN, LAN, etc., using any communication protocol as is well known.

In general, a user accesses and queries proxy server 20, servers 50_1 to 50_N, and other servers through a client 30 to view and download content such as news stories, advertising content, search query results including links to various websites and so on. Such content can also include other media objects such as video and audio clips, URL links, graphic and text objects such as icons and links, and the like. Additionally, such content is typically presented to the user as a web page formatted according to downloaded JavaScript code and HTML code and data as is well known. It will be appreciated that the techniques of the present invention are equally applicable to processing other types of code such as Java code and ActiveX code, and any markup language, including any instance of SGML, such as XML, WML, HTML, DHTML and HDML.

According to one embodiment of the invention, a user preferably accesses servers 50_1 to 50_N through proxy server 20. In the context of electronic commerce, for example, a user may access a local commerce site that provides access (via URL or other selectable links or references) to remote commerce sites, such as individual commerce web sites. One such system is described in U.S. Patent Application Serial Number 09/372,350 (Atty. Docket No. 017887-002500), entitled "Electronic Commerce System for Referencing Remote Commerce Sites at a Local Commerce Site," filed August 11, 1999, the contents of which are hereby incorporated by reference in their entirety for all purposes. As described therein, a Remote Merchant Integration Server (RMIS) provides an interface to multiple merchant web sites. A user that accesses a remote commerce site through the RMI proxy is presented with a slightly modified version of the commerce site by the RMI server. Any requests from the user to a remote commerce site is managed by the RMI server and any responses by the remote commerce site are also managed by the RMI server transparently to both the user and the remote commerce site. One advantage of such a system includes the ability to provide, in the commerce context, a single shopping basket to a user who desires to shop at multiple remote commerce sites. Another advantage is the ability to track transactional information associated with users' purchases at the various merchant sites. An example of such a system can be located on the Internet at the Yahoo! Shopping site (URL: http://shopping.yahoo.com/). In this example, it is desirable to modify JavaScript code and HTML code and data received from the remote commerce sites using the techniques of the present invention to facilitate integration of the RMI system and to maintain user connection to the RMI system during transactions with the remote commerce sites.

According to an embodiment of the present invention, a set of different software morphing agents are provided for handling different kinds of software technologies. For example, one morphing agent (MA) is provided for handling JavaScript and another MA is provided for handling HTML. The MA for each type of the original third-party software technology is delivered to the appropriate device(s), e.g., proxy server 20 and/or a client device 30.

Figure 2 illustrates various exemplary configurations of a runtime environment of a morphing agent (MA) according to embodiments of the present invention. As shown in each configuration, a software source 150, such as a server 50 in Figure 1, provides software to a client device 130 through proxy 120. Depending on the particular MA and its configuration, the software will be modified at the proxy 120, at the client device 130 or partially at the proxy 120 and partially at the client device 130. In configuration a), for example, all components of a particular MA are downloaded to a client device 130 and run in conjunction with a browser application. In configuration b), all components of a particular MA are loaded and run at a proxy server 120. In configuration c), for a particular MA, some components are loaded and run at a proxy server 120 while other components are downloaded to, and run at, a client device 130.

In general, if the code to be modified includes portions that are dynamically assembled, it is preferred that all components for the MA be downloaded to the client device (configuration a). One example of dynamically assembled code in JavaScript could be represented as x + y + "s", where the portion "s" is dynamically assembled or generated by the browser application resident on the client device. Thus, it is preferred that all components of a JavaScript MA be installed on the client side, e.g., at client device 130 to parse and translate dynamically assembled code such as portion "s."

Figure 3 illustrates a general processing flow between modules of a morphing agent according to an embodiment of the present invention. As shown, each morphing agent (MA) includes a tokenizer module 210, a parser module 220 and a translator module 230. Associated with each module is a corresponding rule set, e.g., tokenizer rule set 215, parser rule set 225, and translator rule set 235. An exception handler 240, and associated exception rules set 245, is optionally provided for handling exceptions that occur during the software modification and translation process. Each MA also includes a client-side Runtime Library 250 that includes functions, variable and data configured to run with a browser application. One example of a runtime library can be found in Appendix A.

In operation, before the original software content 260 is processed or executed, all necessary MA components for modifying that particular software type must be installed at the client device 130 (via downloading) and/or at a proxy server 120. The MA then "morphs" (e.g., tokenizes, parses and translates) the original software content 260 into the desired software content 270. Optionally provided exception handler 240 handles any errors that occur during the morphing process. In one embodiment, if an exception, or error, occurs during the morphing process, the exception handler causes the process to be bypassed. The tokenizer 210, parser 220, translator 230, and exception handler 240 are all configurable via their respective rule sets (i.e. 215, 225, 235, 245). The desired output 270 can then work together with the client-side runtime library 250 (via downloading) in a user's browser.

Once the original software content is received, tokenizer module 210 analyzes the string of characters representing a code segment and maps the characters to various token types according to the tokenizer rule set 215. Typical JavaScript token types include string, integer, keyword, identifier, etc. Parser 220 then parses the resulting set of tokens according to the parser rule set 225 to determine relationships between the token types associated with the code segment being analyzed. In one embodiment, a hierarchical tree structure relating the various tokens is created. One example of tokenizer and parser code useful for tokenizing and parsing JavaScript is the NGS JavaScript Interpreter which can be located and downloaded from the Internet at URL: http://www.ngs.fi/js/, the contents of which are hereby incorporated for reference for all purposes. Translator module 230 then transforms the code segment into the desired software content 270 based on the specific translator rule set 235, the token types, and the relationships between the tokens determined by parser 220.

It may be necessary to modify the above NGS JavaScript Interpreter (tokenizer and parser, in particular) to run more efficiently when integrated with a browser application such as Microsoft Internet Explorer or Netscape Navigator. Appendix B includes examples of a modified tokenizer and parser from NGS JavaScript Interpreter, translator code and code for integrating the modified tokenizer and parser with a Browser, according to one embodiment of the present invention.

A typical translator module (230 + 235) of an MA for transforming JavaScript transforms function calls and variables to new function calls and variables to be used together with a client-side runtime library 250 in a user's browser. For example, a function call "open()" is translated according to one embodiment as follows:

"open (URL, TARGET, OPT)" is translated to

"new_function1(URL, TARGET, OPT)", where the function "new_function1()" is implemented in the client-side runtime library.

Similarly, a variable assignment expression is translated according to one embodiment as follows:

"OBJ.location=URL" is translated to

"OBJ.location=new_function2(URL)", where the function "new_function2" is implemented in the client-side runtime library.

Appendix C illustrates examples of translation rules (e.g., 235) for translating function calls and variables according to one embodiment of the present invention.

While the invention has been described by way of example and in terms of the specific embodiments, it is to be understood that the invention is not limited to the disclosed embodiments. To the contrary, it is intended to cover various modifications and similar arrangements as would be apparent to those skilled in the art. Therefore, the scope of the appended claims should be accorded the broadest interpretation so as to encompass all such modifications and similar arrangements.